Limits on the Adaptive Security of Yao's Garbling

Chethan Kamath[1⋆], Karen Klein[2⋆⋆], Krzysztof Pietrzak[2⋆⋆], and Daniel Wichs[3⋆⋆⋆]

[1] ckamath@protonmail.com
[2] IST Austria {kklein,pietrzak}@ist.ac.at
[3] Northeastern University and NTT Research wichs@northeastern.edu

Abstract. Yao's garbling scheme is one of the most fundamental cryptographic constructions. Lindell and Pinkas (Journal of Cryptograhy 2009) gave a formal proof of security in the *selective* setting where the adversary chooses the challenge inputs before seeing the garbled circuit assuming secure symmetric-key encryption (and hence one-way functions). This was followed by results, both positive and negative, concerning its security in the, stronger, *adaptive* setting. Applebaum et al. (Crypto 2013) showed that it cannot satisfy adaptive security as is, due to a simple incompressibility argument. Jafargholi and Wichs (TCC 2017) considered a natural adaptation of Yao's scheme (where the output mapping is sent in the *online phase*, together with the garbled input) that circumvents this negative result, and proved that it is adaptively secure, at least for shallow circuits. In particular, they showed that for the class of circuits of depth δ, the loss in security is at most exponential in δ. The above results all concern the *simulation-based* notion of security.

In this work, we show that the upper bound of Jafargholi and Wichs is basically optimal in a strong sense. As our main result, we show that there exists a family of Boolean circuits, one for each depth $\delta \in \mathbb{N}$, such that *any* black-box reduction proving the adaptive *indistinguishability* of the natural adaptation of Yao's scheme from any symmetric-key encryption has to lose a factor that is exponential in $\sqrt{\delta}$. Since indistinguishability is a weaker notion than simulation, our bound also applies to adaptive simulation.

To establish our results, we build on the recent approach of Kamath et al. (Eprint 2021), which uses pebbling lower bounds in conjunction with oracle separations to prove fine-grained lower bounds on loss in cryptographic security.

⋆ Most of the work was done while the author was at Northeastern University, supported by the IARPA grant IARPA/2019-19-020700009, and Charles University, funded by project PRIMUS/17/SCI/9.

⋆⋆ Funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (682815 - TOCNeT).

⋆⋆⋆ Research supported by NSF grant CNS-1750795 and the Alfred P. Sloan Research Fellowship.

1 Introduction

A garbling scheme allows one to garble a circuit C and an input x such that only the output $\mathsf{C}(x)$ can be learned while everything else – besides some leakage such as the size or topology of the circuit – remains hidden. It was originally used by Yao as a means to achieve secure function-evaluation [17,18]. Despite its huge impact on cryptography, it was formally defined as a stand-alone primitive only much later by Bellare, Hoang and Rogaway [6]. In addition to a syntactic definition, they propose two different security notions for garbling schemes: simulatability and indistinguishability. They show the equivalence of the two definitions[4] in the presence of a *selective* adversary, which sends the circuit and input to be garbled in one shot. In contrast, for the more general case in which the adversary first – in an *offline* phase – chooses a circuit C and then (after receiving its garbling) – in the *online* phase – *adaptively* chooses its input x, the notion of indistinguishability turns out to be strictly weaker than simulatability. Many applications require security in such an adaptive setting, and for the sake of efficiency the cost during the online phase is to be kept minimal.

Prior work on security. Whilst there exist several constructions of provably-secure (even in the adaptive sense) garbling schemes (see Section 1.3), a feature of Yao's scheme (and variants thereof) is that security can be proven under the minimal assumption of one-way functions. At the same time, this scheme offers almost-optimal online complexity, with the size of the garbled input being linear in the input-size, and independent of the output- as well as circuit-size. A formal security proof of Yao's scheme in the *selective* setting was given by Lindell and Pinkas [16]. There exists a generic approach to reduce adaptive security to selective security: the adaptive reduction simply guesses the input x and then runs the selective reduction on the adaptive adversary. This, unfortunately, leads to a loss in security that is exponential in $|x|$. Furthermore, Applebaum et al. [3] showed that the *online complexity* of any adaptively-simulatable garbling scheme must exceed the output-size of the circuit, thereby proving a first limitation of Yao's scheme.

All of this led Jafargholi and Wichs [14] to consider a natural adaptation of Yao's garbling scheme (described in Section 1.1), where the mapping of output labels to output bits is sent in the online phase as part of the garbled input (see below for the construction). The negative result by Applebaum et al. does not apply to this adaptation of Yao's garbling scheme since its online complexity exceeds the output size. Therefore, this adaptation is the natural version of Yao's garbling scheme for the case of adaptive security, and is the scheme that we consider in this work and will simply refer to as "Yao's garbling" from now on. Jafargholi and Wichs [14] were able to show that it satisfies adaptive security for a wide class of circuits, including $\mathbf{NC}^1$ circuits. More precisely, they

[4] In the security game for simulatability, the simulator has to simulate $\tilde{\mathsf{C}}$ given only the output $y = \mathsf{C}(x)$ and some *leakage* $\Phi(\mathsf{C})$. While equivalence of selective simulatability and selective indistinguishability holds for the most natural leakage functions (e.g. the size or topology of C), it *does not* hold for arbitrary leakage functions Φ.

prove adaptive security of Yao's garbling via a black-box reduction to the IND-CPA security of the underlying symmetric-key encryption (SKE) scheme with a loss in security that is exponential in the *depth* of the circuit. Their proof employs a specially tailored *pebble game* on graphs, and is an application of the *piecewise-guessing framework* of Jafargholi et al. [11]. Since our work concerns the optimality of this proof, let's look at it in a bit more detail.

1.1 Yao's Scheme and Adaptive Indistinguishability

Let's first informally recall Yao's garbling scheme. A circuit $\mathsf{C} : \{0,1\}^n \to \{0,1\}^\ell$ is garbled in the offline phase as follows:

1. For each wire w in C, choose a pair of secret keys $k_w^0, k_w^1 \leftarrow \mathsf{Gen}(1^\lambda)$ for a SKE $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
2. For every gate $g : \{0,1\} \times \{0,1\} \to \{0,1\}$ with left input wire u, right input wire v, and output wire w, compute a garbling table $\tilde{\boldsymbol{g}}$ consisting of the following four ciphertexts (in a random order).

$$\begin{aligned} c_1 &:= \mathsf{Enc}_{k_u^0}(\mathsf{Enc}_{k_v^0}(k_w^{g(0,0)})) \qquad & c_2 &:= \mathsf{Enc}_{k_u^1}(\mathsf{Enc}_{k_v^0}(k_w^{g(1,0)})) \\ c_3 &:= \mathsf{Enc}_{k_u^0}(\mathsf{Enc}_{k_v^1}(k_w^{g(0,1)})) \qquad & c_4 &:= \mathsf{Enc}_{k_u^1}(\mathsf{Enc}_{k_v^1}(k_w^{g(1,1)})) \end{aligned} \tag{1}$$

3. If C has s wires and output wires denoted by $w_{s-\ell+1}, \ldots, w_s$, assemble the output mapping $\{k_w^b \to b\}_{i \in [s-\ell+1,s],\, b \in \{0,1\}}$.

The garbled circuit $\tilde{\mathsf{C}}$ consists of all the garbling tables $\tilde{\boldsymbol{g}}$ as well as the output mapping. To garble an input $x = (b_1, \ldots, b_n)$ in the online phase, simply set

$$\tilde{x} := (k_{w_1}^{b_1}, \ldots, k_{w_n}^{b_n})$$

where w_i denotes the ith input wire. The only difference in the variant from [14] is that the sending of the output mapping is moved to the online phase, which leads to an increase in the online complexity to linear in the input- *and* output-size.

To evaluate the garbled circuit on the garbled input, one requires the following *special property* of the SKE: For each ciphertext $c \leftarrow \mathsf{Enc}_k(m)$ there exists a unique key – namely k – such that decryption doesn't fail. Evaluation of the garbled circuit given the garbled input then works starting from the gates at the lowest level by simply trying which of the four ciphertexts can be decrypted using the two given input keys. This allows to recover exactly one of the two keys associated to the output wire of the respective gate and in the end the output mapping is used to map the sequence of revealed output keys to an output string $y \in \{0,1\}^\ell$.

Adaptive indistinguishability. A garbling scheme is adaptively indistinguishable if no efficient adversary can succeed in the following experiment[5] with non-negligible advantage:

[5] In fact, we define a *weaker* security notion than indistinguishability as defined in [6]; according to their definition the adversary can choose *two* circuits $\mathsf{C}_0, \mathsf{C}_1$ of the same

1. The adversary submits a circuit C to the challenger, who responds with $\tilde{\mathsf{C}}$.
2. The adversary then submits a pair of inputs (x_0, x_1).
3. The challenger flips a coin b and responds with $\tilde{x}_b$.
4. The adversary wins if it guesses the bit b correctly.

In the following, we will refer to the two games for $b = 0$ and $b = 1$ as the "left" and "right" games, respectively.

To prove adaptive indistinguishability[6] of Yao's scheme for an arbitrary SKE (satisfying the special property), Jafargholi and Wichs construct a black-box reduction from the IND-CPA security of the SKE. More precisely, they proceed by a hybrid argument, where they define a sequence of hybrid games interpolating between the left and the right game such that each pair of subsequent hybrid games only differs in a single ciphertext (in the garbling table) and can be proven indistinguishable by relying on the IND-CPA security of the SKE.

The loss in security incurred by such a reduction then depends on the length of the sequence and the amount of information required to simulate the hybrid games. To end up with a meaningful security guarantee, thus, the sequence of hybrid games must not be too long and it must be possible to simulate any of the hybrid games without relying on too much information, particularly the knowledge of the entire input. Jafargholi and Wichs design such a sequence of hybrid games by using an appropriate pebble game on the topology graph underlying the circuit. In that game, a pebble on a gate indicates that the gate is not honestly garbled (as in Equation (1)) but is, instead, garbled in some *input-dependent* mode. The pebble rules, which dictate when a pebble can be placed on or removed from a vertex, guarantee that two subsequent hybrids can be proven indistinguishable, and the loss in security directly relates to the number of pebbles on the graph.

Keeping this proof technique in mind, the main idea of this work is to turn a pebble lower bound (w.r.t. an appropriate pebble game) into a lower bound on the security loss inherent to any black-box reduction of adaptive indistinguishability of Yao's scheme. Such an approach was recently adopted by Kamath et al. [15], also in the context of adaptive security but for primitives that are of a different flavour (e.g., multi-cast encryption). However, the case of garbled circuits turns out very different for several reasons we will highlight later (see Section 2.5).

1.2 Our Results

We prove a lower bound on the loss in security incurred by any black-box reduction proving adaptive indistinguishability of Yao's garbling scheme [14] from IND-CPA security of the SKE scheme. This immediately implies a similar lower

topology and inputs x_0, x_1 such that $\mathsf{C}_0(x_0) = \mathsf{C}_1(x_1)$. Aiming at a lower bound on the gap between the security of Yao's scheme and the security of the underlying SKE, the additional restriction we put on our adversary only strengthens our results.

[6] To be precise, [14] prove the stronger security notion of simulatability, which implies indistinguishability.

bound with respect to the (stronger) more common security notion of adaptive simulatability. Our lower bound is subexponential in the depth d of the circuit, hence almost matches the best known upper bound from [14].

Theorem (main, Theorem 4.1). Any black-box reduction from adaptive indistinguishability (and thus also simulatability) of Yao's garbling scheme on the class of circuits with input length n and depth $\delta \leq 2n$ to the IND-CPA security of the underlying SKE loses at least a factor $\mathsf{loss} = \frac{1}{q} \cdot 2^{\sqrt{\delta}/61}$, where q denotes the number of times the reduction rewinds the adversary.

Two remarks concerning the theorem are in order. Firstly, we are proving a negation of the statement in [14], which upper bounds loss for *every* graph in a class. Therefore, when we say that the class of circuits above loses at least a factor loss, we mean that there *exists* some circuit G in that class such that any reduction loses by that factor (and not that every circuit in that class loses by that factor). The design of this circuit G is one of the main technical contributions of this work. The second remark concerns the design of this circuit G. In addition to some structural properties that we will come to later, we design G to output the constant bit 0. This implies that the output mapping can easily be guessed by a reduction, and therefore the difference, in this case, between Yao's original scheme and [14] is only marginal.

Comparison with Applebaum et al. [3]. The result in [3] rules out adaptively-simulatable randomised encodings with online complexity less than the output-size of the function it encodes. Since Yao's garbling is one instantiation of randomised encodings, their result immediately rules out its adaptive simulatability. However, [3] does not apply to our setting for three reasons. Firstly, their result only applies to the original construction of Yao's garbled circuits where the garbled input can be smaller than the output size. In this work we consider the adaptation of Yao's garbling scheme [14] where the output mapping is sent in the online phase, hence the online complexity always exceeds the output size. Secondly, their result applies to circuits with large output, while our result holds even for Boolean circuits with outputs of length 1. Finally, their result only applies to simulation security, while our result even holds for indistinguishability.

Comparison with Hemenway et al. [10]. We would like to emphasise that our lower bound only holds for the *specific* construction of Yao's garbled circuits, and it does not rule out other constructions, even potentially from one-way functions. In fact, the construction of Hemenway et al. already circumvents our result and it is instructive to see how. On a high level, their idea (similar to [5]) is to take Yao's garbling scheme and then encrypt all the resulting garbling tables with an additional layer of "somewhere equivocal" encryption on top. This change allows them to prove adaptive security with only a *polynomial* loss in security (at the cost of increased online complexity). The intuitive reason why our approach does not apply to this construction is that the additional layer of encryption somehow "blurs out" all the details about the individual garbling tables, on which our argument depends (see Section 2.4).

1.3 Further Related Work on Adaptive Security

Adaptive security for garbled circuits. The problem of constructing adaptively-secure garbling schemes was first raised by Bellare, Hoang and Rogaway in [5]; they gave a first adaptively-secure construction in the random oracle model, which bypasses the lower bound of Applebaum et al. [3]. Bellare, Hoang and Keelveedhi [4] then proved the previous scheme adaptively-secure in the standard model, but under non-standard assumptions on hash functions. Further constructions from various assumption followed: Boneh et al. [7] constructed an adaptively-secure scheme from the learning with errors (LWE) assumption, where the online complexity depends on the depth of the circuit family. Ananth and Sahai [2] constructed an optimal garbling scheme from iO. In [13], Jafargholi et al. relax the simulation-based security to *indistinguishability* and show how to construct adaptively-secure garbling schemes from the minimal assumption of one-way functions, where the online complexity only depends on the pebble complexity and the input-size, but is *independent* of the output-size. Later, Ananth and Lombardi [1] constructed succinct garbling schemes from functional encryption. A particularly strong result in this area was due to Garg and Srinivasan [9], who constructed adaptively-secure garbling with near optimal online complexity that can be based on standard assumptions such as the computational Diffie-Hellman (CDH), the factoring, or the LWE assumption. While this list is far from complete, we finally mention a recent work by Jafargholi and Oechsner [12] who analyze adaptive security of several practical garbling schemes. They give positive as well as negative results, and argue why the techniques from [14] cannot be applied to certain garbling schemes.

Adaptive security for other graph-based games. Jafargholi et al. gave a framework for proving adaptive security [11], also known as *piecewise guessing* technique. Beside several applications to other *graph-based* security games, this framework also comprises the reduction from [14] as a special case. Kamath et al. [15] considered optimality of this approach for certain graph-based games which arise in the context of e.g., multicast encryption, continuous group key agreement, and constrained PRF. They gave non-trivial fine-grained lower bounds on the loss in adaptive security incurred by (oblivious) reductions via *pebble lower bounds.*

2 Technical Overview

We aim to prove *fine-grained* lower bounds on loss in security incurred by black-box reductions in a setting where a primitive F is used in a protocol Π^F. In our case F is SKE and Π^F is Yao's garbling scheme using the SKE. In order to bound loss, the loss in security incurred by any efficient black-box reduction R that breaks F when given black-box access to an adversary that breaks Π^F (i.e., from F to Π^F), we must show that for *every* R, there exists

- an instance F (not necessarily efficiently-implementable) of F and
- an adversary A (not necessarily efficient) that breaks Π^{F}

such that loss in security incurred by R in breaking F is at least loss.[7] We next describe how the instance and the adversary are defined in our setting.

2.1 Our Oracles

We define two oracles $\mathcal{F}$ and $\mathcal{A}$ implementing an ideal SKE and an adversary, respectively, such that

- the SKE scheme $\mathcal{F} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies IND-CPA security *information-theoretically*,
- the (inefficient) adversary $\mathcal{A}$ breaks indistinguishability of the garbling scheme $\Pi^{\mathcal{F}}$, but is not helpful in breaking the IND-CPA security of $\mathcal{F}$.

Ideal encryption. We will define the ideal SKE oracle $\mathcal{F}$ such that Enc is defined through a random expanding function (which is injective with overwhelming probability). Since the security of $\mathcal{F}$ is information-theoretic, any advantage against IND-CPA which a reduction with oracle access to $\mathcal{F}$ and $\mathcal{A}$ obtains must stem (almost) entirely from the interaction with $\mathcal{A}$. This is true since the reduction can only make polynomially many queries and thus the probability that the answer to one of its oracle queries coincides with the IND-CPA challenge is negligible. On the other hand, a computationally unbounded adversary using an unlimited number of queries can break the scheme and (thanks to injectivity) perfectly recover messages and secret keys from any ciphertext.

The adversary. As for the (inefficient) adversary $\mathcal{A}$, we define a so-called *threshold* adversary which does the following in the indistinguishability game:

1. $\mathcal{A}$ chooses a particular circuit G (see Section 2.3) which has constant output (bit) 0 and sends G to the challenger.
2. After receiving the garbled circuit $\tilde{\mathsf{G}}$, $\mathcal{A}$ chooses garbling inputs x_0 and x_1 uniformly at random and sends them to the challenger. Note that $\mathsf{G}(x_0) = \mathsf{G}(x_1)$ trivially holds since G has constant output.
3. On receipt of the garbled input $\tilde{x}_b$ along with an output mapping, $\mathcal{A}$ first runs some initial checks on $(\tilde{\mathsf{G}}, \tilde{x}_b)$ to verify that the garbling has the correct syntax, and then *extracts a pebble configuration* $\mathcal{P}$ on G (see Section 2.4). That is, every gate in G is either assigned a pebble or not, depending on the content of its garbling table in $\tilde{\mathsf{G}}$ and the garbled input $\tilde{x}_b$. To compute this mapping, the inefficient adversary $\mathcal{A}$ simply breaks the underlying encryption by brute force. Finally, $\mathcal{A}$ outputs 0 (denoting 'left') if the extracted pebble configuration is *good* (defined later through some pebble game), and 1 (denoting 'right') otherwise.

[7] This is obtained by simply negating the definition of a black-box reduction: *there exists an efficient reduction* R *for every implementation (not necessarily efficient)* F *of* F *and for every (not necessarily efficient) adversary* A *that breaks* Π^{F} *such that the loss in security is at most* loss.

By design, the left indistinguishability game (where $b = 0$) will correspond to a good configuration, whereas the right game will not. Therefore the above adversary is a valid distinguisher for the indistinguishability game (Lemma 4.5). Moreover, $\mathcal{A}$ concentrates all its distinguishing advantage at the *threshold* of good and bad configurations (hence the name). Therefore, intuitively speaking, for any reduction to exploit $\mathcal{A}$'s distinguishing advantage, it must somehow embed its own (IND-CPA) challenge at the threshold. All the technicality in proving our main theorem goes into formalising this intuition, which we summarise next in Section 2.2.

2.2 High-Level Idea

To prove a lower bound on loss (Theorem 4.1), we construct a *punctured* adversary $\mathcal{A}[c^*]$ (see Section 4.5) which behaves similar to $\mathcal{A}$ *except* when it comes to the hardcoded challenge ciphertext $c^* \leftarrow \mathsf{Enc}_{k^*}(m)$ (for some arbitrary message m). We aim to puncture $\mathcal{A}[c^*]$ such that it never decrypts c^* but instead just proceeds by assuming that c^* decrypts to the all-0 string, and hence cannot be of any help to a reduction that aims to break c^*. However, we have to be careful here since the reduction embedding c^* in $\tilde{\mathsf{G}}$ will also embed other ciphertexts under key k^* (which it can derive through querying its IND-CPA encryption oracle Enc_{k^*}), and hence $\mathcal{A}[c^*]$ would learn the key k^* when brute-force decrypting these ciphertexts. We solve this issue by endowing $\mathcal{A}[c^*]$ with a decryption oracle Dec_{k^*} that allows to find and decrypt those ciphertexts under k^*. Since our ideal encryption scheme actually satisfies the stronger notion of IND-CCA security, this decryption oracle is of no help to the reduction.

The core of our lower bound is now to define the circuit G and the notion of *good* pebble configurations such that the following holds:

- Our threshold adversary $\mathcal{A}$ indeed breaks the garbling scheme.
- It is hard to distinguish $\mathcal{A}$ from $\mathcal{A}[c^*]$.

For the latter property, note that any efficient reduction R can only distinguish $\mathcal{A}$ from $\mathcal{A}[c^*]$ if their outputs differ, which only happens if they extract different pebbling configurations $\mathcal{P} \neq \mathcal{P}^*$ such that one of them is good and the other bad. Thus, to bound the success probability of R, it suffices to establish the following two properties:

1. The pebbling configurations $\mathcal{P}$ and $\mathcal{P}^*$ extracted by $\mathcal{A}$ and $\mathcal{A}[c^*]$ (in the same execution of the game, using the same randomness) differ by at most one valid pebbling move in some natural pebble game[8], where a pebble can be placed on or removed from a gate if at least one of its parent gates carries a pebble.

[8] In Section 4.3 we actually consider a much more finegrained pebble game, where different types of pebbles represent different garbling modes of a gate. For this exposition, it suffices to focus on this simplified game.

2. It is hard for any reduction to produce $(\tilde{\mathsf{G}}, \tilde{x})$ such that $\mathcal{A}$ extracts a *threshold* configuration, i.e. a pebble configuration that is good but can be switched to a bad configuration within one valid pebbling move.

Intuitively, pebbles on gates in the circuit represent malformed gates, i.e., gates whose garbling table is different from the honest garbling table. When considering circuits consisting only of non-constant gates, the pebbling rule in Property 1 captures the fact that a reduction cannot produce ciphertexts encrypting the key k^* under which its challenge ciphertext $c^* \leftarrow \mathsf{Enc}_{k^*}(m)$ (for some arbitrary m) was encrypted. Hence, in order to embed c^* at a gate, the reduction has to first output a malformed garbling (not encoding k^*) for its predecessor gate. Now, to see why Property 1 holds – i.e., the pebbling configurations $\mathcal{P}$ and $\mathcal{P}^*$ extracted by $\mathcal{A}$ and $\mathcal{A}[c^*]$ follow the same dynamics – note that the behaviour of $\mathcal{A}$ and $\mathcal{A}[c^*]$ can only differ if k^* is not encrypted in any ciphertext.

The tricky part of our proof is to establish Property 2 which, on a high level, works as follows. For a reduction R to simulate a threshold configuration we first force it to maul – and hence pebble – several gates. Then, for this mauling to go 'undetected' we force R to correctly guess the value of these gates when G is evaluated at x_0. This, intuitively, will be the source of its loss. To this end, we design our circuit G to consist of two blocks[9], $\mathsf{G}^{\oplus}$ and $\mathsf{G}^{\wedge}$. Looking ahead, whether there is a pebble on a gate in $\mathsf{G}^{\oplus}$ will be *independent* of the input and correspond to R's attempt at guessing x_0 (this relies on the properties of XOR gates). The pebbles on $\mathsf{G}^{\wedge}$, in contrast, will be extractable with respect to the input garbling $\tilde{x}_b$ and indicate whether or not the guesses on x_0 in the $\mathsf{G}^{\oplus}$ block were correct (this relies on the properties of AND gates). Moreover, by definition:

– In case of a proper garbling of (G, x_0) (i.e., the left game), the adversary $\mathcal{A}$ will not extract any pebble on $\mathsf{G}^{\oplus}$ or $\mathsf{G}^{\wedge}$.
– In case of a proper garbling of (G, x_1) (i.e., the right game), on the other hand, the adversary $\mathcal{A}$ will not extract any pebbles on $\mathsf{G}^{\oplus}$, but will extract some pebbles on $\mathsf{G}^{\wedge}$ (since $x_1 \neq x_0$).

Accordingly, we define the *good* predicate such that the empty configuration is good, whereas any configuration containing a pebble on $\mathsf{G}^{\wedge}$ is bad, and therefore the above ensures that $\mathcal{A}$ breaks the security of the garbling scheme. Furthermore, the threshold configurations contain many pebbles on $\mathsf{G}^{\oplus}$, but no pebbles on $\mathsf{G}^{\wedge}$. In other words, threshold configurations require R to make many guesses about x_0 and all of them need to be correct, which is unlikely to occur. This establishes Property 2.

2.3 The Circuit G and the Good Predicate

The design of topology of the circuit $\mathsf{G}^{\oplus}$ is such that it has high pebbling complexity with respect to our pebble game: i.e., every valid pebbling sequence

[9] For this high-level overview, we ignore the third block G^{0} consisting of a binary tree of AND gates, whose sole purpose is to guarantee constant 0 (bit) output.

starting from the *initial* empty configuration and reaching a *final* configuration that has a pebble on an output gate of $\mathsf{G}^{\oplus}$, must contain a "heavy" configuration with many, say d, pebbles. To guarantee that threshold configurations contain many pebbles, we define the good configurations as those that are reachable with $d-1$ pebbles following valid pebbling moves. Since $\mathsf{G}^{\wedge}$ will (topologically) succeed $\mathsf{G}^{\oplus}$ in G, any configuration with a pebble on $\mathsf{G}^{\wedge}$ is in particular bad (since an output gate of $\mathsf{G}^{\oplus}$ must have been pebbled first). At the same time, to allow for our "control mechanism", we construct G so that each gate g in $\mathsf{G}^{\oplus}$ has a 'companion' successor gate in $\mathsf{G}^{\wedge}$ that helps check correctness of g's output. Thus for each AND gate in $\mathsf{G}^{\wedge}$, one of the inputs comes from the output of $\mathsf{G}^{\oplus}$ and the other from the output of its companion gate (see Figure 1). This fixes the topology of G and we choose the type of gate as to enforce Property 2, as explained below.

- The $\mathsf{G}^{\oplus}$ circuit is composed only of XOR gates, since these gates allow us to maintain *high entropy* (of the input), and hence guarantee that it is hard to guess the outputs of the pebbled gates in $\mathsf{G}^{\oplus}$ (see Section 4.2). Furthermore, XOR gates are *symmetric* with respect to their input in the sense that from the garbling table alone even an inefficient adversary cannot distinguish which keys are associated with which bits. This property allows $\mathcal{A}$ to extract the pebbling configuration of $\mathsf{G}^{\oplus}$ just from $\tilde{\mathsf{G}}$, independently of the input (see next section).
- The $\mathsf{G}^{\wedge}$ circuit, on the other hand, is composed of AND gates. Since AND gates are *asymmetric* (since only $(1,1)$ maps to 1, while all three other input pairs map to 0), we can use them to detect errors in the $\mathsf{G}^{\oplus}$ circuit: i.e., looking at a garbling table of an AND gate our adversary $\mathcal{A}$ can exploit this asymmetry to easily associate keys to bits. Thus, whenever during evaluation of $\tilde{\mathsf{G}}$ on input $\tilde{x}$ the adversary $\mathcal{A}$ receives wrong input keys for a (properly garbled) AND gate, $\mathcal{A}$ considers this gate as malformed and associates it with a pebble. (The case of AND gates which are not properly garbled is rather technical and we refer the reader to Section 4.4.)

2.4 Extracting the Pebble Configuration

Since it is central to the working of our adversary $\mathcal{A}$ (and is a somewhat subtle matter), here we provide a high-level description of the extraction mechanism.[10] First of all, recall that pebbles on $\mathsf{G}^{\oplus}$ and $\mathsf{G}^{\wedge}$ have different meanings: a pebbled XOR gate indicates that its garbling table is *malformed* whereas a pebbled AND gate indicates that R's guess for the companion XOR gate is *wrong*. This, coupled

[10] In Section 4.4 we consider a more general extraction mechanism that can be extended to arbitrary gates and assigns different types of pebbles, representing the "distance" of a garbling table $\tilde{\boldsymbol{g}}'$ for a gate g from an honest garbling table $\tilde{\boldsymbol{g}}$. For ease of exposition, here we consider a simplified pebble game and only discuss how to extract pebbles for XOR and AND gates, where a pebble in this simplified game would correspond to different sets of pebbles for XOR and AND gates in the more fine-grained pebble game.

with the fact that the gates have differently-structured gate tables (i.e., symmetric vs. asymmetric) means that the extraction mechanism for the two gates (and hence the blocks) is also different. In particular, as we will see, the pebble status of an XOR gate is something that can inferred solely from the garbled circuit $\tilde{\mathsf{G}}$ (and thus can be done in the offline phase) whereas the pebble status of an AND gate is something that also depends on the garbled input $\tilde{x}$ and is necessarily done in the online phase. Let's look at how the respective extraction is carried out. First, given $\tilde{\mathsf{G}}$, $\mathcal{A}$ extracts a key pair for each wire in G from the encryptions associated with its *successor* gates, or the output mapping; if this cannot be done uniquely, $\mathcal{A}$ aborts and outputs 1 (we refer to Section 4.4 for more details). In the following, for a gate g, let u and v denote the input wires, w the output wire, and k_u, k'_u, k_v, k'_v, k_w, k'_w the corresponding keys associated with these wires.

– If g is an XOR gate, then the honest garbling table of g can be derived from Equation (1) as

$$\begin{array}{ll} \mathsf{Enc}_{k_u}(\mathsf{Enc}_{k_v}(k_w)) & \mathsf{Enc}_{k'_u}(\mathsf{Enc}_{k_v}(k'_w)) \\ \mathsf{Enc}_{k_u}(\mathsf{Enc}_{k'_v}(k'_w)) & \mathsf{Enc}_{k'_u}(\mathsf{Enc}_{k'_v}(k_w)). \end{array}$$

  Whenever a garbling table $\tilde{\boldsymbol{g}}$ differs from this representation (i.e., not symmetric), $\mathcal{A}$ assigns g a pebble and this assignment is *independent* of the bits running over the wires u, v, w and the keys revealed during evaluation. Thus, $\mathcal{A}$ can extract pebbles on $\mathsf{G}^{\oplus}$ already *before* it chose the inputs x_0, x_1, in particular independently of $\tilde{x}$.
– For an AND gate g, on the other hand, the garbling table of g consists of four ciphertexts derived from Equation (1) as

$$\begin{array}{ll} \mathsf{Enc}_{k_u}(\mathsf{Enc}_{k_v}(k_w)) & \mathsf{Enc}_{k'_u}(\mathsf{Enc}_{k_v}(k_w)) \\ \mathsf{Enc}_{k_u}(\mathsf{Enc}_{k'_v}(k_w)) & \mathsf{Enc}_{k'_u}(\mathsf{Enc}_{k'_v}(k'_w)). \end{array}$$

  Since the roles of the keys are *asymmetric*, the pebble extraction will depend on the bits b_u, b_v, b_w running over the wires and the keys k^r_u, k^r_v, k^r_w revealed during evaluation. A first attempt would be to simply map keys to bits as $k_u, k_v, k_w \to 0$ and $k'_u, k'_v, k'_w \to 1$, and assign g a pebble if $k^r_\eta \not\to b_\eta$ for some $\eta \in \{u, v, w\}$. Unfortunately, this simple idea does not work since a reduction R might embed its challenge ciphertext $c^* \leftarrow \mathsf{Enc}_{k^*}(m)$ in the garbling of an AND gate (recall from Section 2.3 that the gates in $\mathsf{G}^{\wedge}$ receive one input from an output gate of $\mathsf{G}^{\oplus}$ and the other input from their companion gate *within* the circuit $\mathsf{G}^{\oplus}$). Now, if R embeds the challenge key k^* at an output wire of $\mathsf{G}^{\oplus}$, it must pebble an output gate in $\mathsf{G}^{\oplus}$, hence end up with a bad pebbling configuration independently of c^*. However, this is not true if R embeds k^* at the other input wire of the AND gate. Thus, $\mathcal{A}$ must not extract a pebble for a garbling table that can be derived from an honest garbling table by embedding a challenge key at this wire. We show in Section 4.4 that such malformed garblings of AND gates either involve guessing the input bits or they can still be used for our "control mechanism".

2.5 Comparison with [15]

While both, [15] and our work, model choices made by a reduction by putting pebbles on a graph structure, the analogy basically ends there. In [15] an interactive game between a "builder" and a "pebbler" is considered in which the builder chooses edges and the pebbler decides adaptively whether to pebble them. The goal of the pebbler is to get into a "good" configuration, and the difficulty for the reduction (playing the role of the pebbler) there lies in the fact that the graph is only revealed edge-by-edge. In contrast, in this work the graph structure is initially known and the game has just two rounds. The difficulty for the reduction here comes from having to guess the bits running over a subset of wires during evaluation of the circuit. None of the main ideas from [15] seem applicable in this setting and vice versa. For example, most of the results in [15] are restricted to the limited class of so-called oblivious reductions, while our setting doesn't share the difficulties encountered in [15]; in particular, our result holds for arbitrary black-box reductions.

3 Preliminaries

Notation and Definitions. For integers $m, n \in \mathbb{N}$ with $m < n$, let $[n] := \{1, 2, \ldots, n\}$, $[n]_0 := \{0, 1, \ldots, n\}$, and $[m, n] := \{m, m+1, \ldots, n\}$. For two sets $\mathcal{S}, \mathcal{S}'$ we write $\mathcal{S} \subset \mathcal{S}'$ if $\mathcal{S}$ is a (not necessarily strict) subset of $\mathcal{S}'$. Furthermore, let log be always base 2. For the classical definitions of IND-CPA and IND-CCA security of symmetric-key encryption (SKE) we refer the reder to the full version of this paper.

Garbling schemes. The definitions are taken mostly from [13]; more details can be found in [6].

Definition 3.1. *A garbling scheme* **GC** *is a tuple of PPT algorithms* $(\mathsf{GCircuit}, \mathsf{GInput}, \mathsf{GEval})$ *with syntax and semantics defined as follows.*

$(\tilde{\mathsf{C}}, K) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{C})$**.** *On inputs a security parameter* λ *and a circuit* $\mathsf{C} : \{0,1\}^n \rightarrow \{0,1\}^\ell$*, the* garble-circuit *algorithm* $\mathsf{GCircuit}$ *outputs the* garbled circuit $\tilde{\mathsf{C}}$ *and* key K*.*

$\tilde{x} \leftarrow \mathsf{GInput}(K, x)$**.** *On input an input* $x \in \{0,1\}^n$ *and key* K*, the* garble-input *algorithm* GInput *outputs* $\tilde{x}$*.*

$y = \mathsf{GEval}(\tilde{\mathsf{C}}, \tilde{x})$**.** *On input a garbled circuit* $\tilde{\mathsf{C}}$ *and a garbled input* $\tilde{x}$*, the* evaluate *algorithm* GEval *outputs* $y \in \{0,1\}^\ell$*.*

Correctness. There is a negligible function $\epsilon = \epsilon(\lambda)$ such that for any $\lambda \in \mathbb{N}$, any circuit C and input x it holds that

$$\Pr\left[\mathsf{C}(x) = \mathsf{GEval}(\tilde{\mathsf{C}}, \tilde{x})\right] = 1 - \epsilon(\lambda),$$

where $(\tilde{\mathsf{C}}, K) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{C})$, $\tilde{x} \leftarrow \mathsf{GInput}(K, x)$.

In this work we only consider the security notion of adaptive indistinguishability. For reference we provide the definition of the strictly stronger notion of adaptive simulatability in the full version of this paper.

Definition 3.2 (Adaptive Indistinguishability). *A garbling scheme* $\mathbf{GC}$ *is* (ϵ, T)*-adaptively-indistinguishable for a class of circuits* $\mathcal{C}$*, if for any probabilistic adversary* A *of size* $T = T(\lambda)$*,*

$$\left|\Pr\left[\mathsf{Game}_{\mathsf{A},\mathbf{GC}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Game}_{\mathsf{A},\mathbf{GC}}(1^\lambda, 1) = 1\right]\right| \leq \epsilon(\lambda).$$

where the experiment $\mathsf{Game}_{\mathsf{A},\mathbf{GC},\mathsf{S}}(1^\lambda, b)$ *is defined as follows:*

1. A *selects a circuits* $\mathsf{C} \in \mathcal{C}$ *and receives* $\tilde{\mathsf{C}}$*, where* $(\tilde{\mathsf{C}}, K) \leftarrow \mathsf{GCircuit}(1^\lambda, \mathsf{C})$*.*
2. A *specifies* x_0, x_1 *such that* $\mathsf{C}(x_0) = \mathsf{C}(x_1)$ *and receives* $\tilde{x}_b \leftarrow \mathsf{GInput}(x_b, K)$*.*
3. *Finally,* A *outputs a bit* b'*, which is the output of the experiment.*

In the indistinguishability game as defined in [6] the adversary can select *two* circuits $\mathsf{C}_0, \mathsf{C}_1$ of the same topology and receives a garbling $\tilde{\mathsf{C}}_b$ of one of them. The choice of input x_0, x_1 is then restricted to satisfy $\mathsf{C}_0(x_0) = \mathsf{C}_1(x_1)$. Our notion of indistinguishability is clearly weaker, which strengthens our lower bound.

Yao's garbled circuit. In the full version of this paper we describe the variant [14] of Yao's garbling scheme $\Pi^{\mathcal{F}}$ based on a symmetric encryption scheme $\mathcal{F}$ with the special property defined below. Recall that in contrast to the original scheme, here the output map is sent along with the garbled input in the online phase.

Definition 3.3 (Special Property of Encryption). *We say an encryption scheme* $\mathcal{F} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfies the special property if for every security parameter* λ*, every key* $k \leftarrow \mathsf{Gen}(1^\lambda)$*, every message* $m \in \mathcal{M}$*, and encryption* $c \leftarrow \mathsf{Enc}_k(m)$ *it holds* $\mathsf{Dec}_{k'}(c) = \perp$ *for all* $k' \neq k$*.*

4 Lower bound for Yao's Garbling Scheme

Let Π denote the variant of Yao's garbling scheme as analysed in [14]. As explained in the introduction, we follow the approach in [15] and define two oracles $\mathcal{F}$ and $\mathcal{A}$ implementing an ideal SKE scheme and an adversary, respectively, such that $\mathcal{A}$ is not helpful in breaking IND-CPA security of $\mathcal{F}$. For the precise description of $\mathcal{F}$ we refer to Section 4.5. The (inefficient) threshold adversary $\mathcal{A}$ we define as follows:

1. On input the security parameter in unary, 1^λ, the adversary $\mathcal{A}$ chooses a circuit G with input size $n = \Theta(\lambda)$, constant output, and depth $\delta(d) \in O(n)$ for a parameter d. The circuit G consists of three parts, i.e., $\mathsf{G} = \mathsf{G}^0 \circ \mathsf{G}^\wedge \circ \mathsf{G}^\oplus$; see introduction. $\mathcal{A}$ sends G to the challenger.

2. After receiving $\tilde{\mathsf{G}}$, the adversary $\mathcal{A}$ chooses $x_0, x_1 \leftarrow \{0,1\}^n$ uniformly at random. Note that $\mathsf{G}(x_0) = \mathsf{G}(x_1)$ trivially holds since G has constant output. $\mathcal{A}$ sends x_0, x_1 to the challenger.
3. On receipt of $\tilde{x}_b = (k_1, \ldots, k_n)$ along with an output mapping, $\mathcal{A}$ extracts a *pebbling configuration* on the graph $G \setminus G^0$ corresponding to $\mathsf{G}^\wedge \circ \mathsf{G}^\oplus$ as described in Section 4.4. $\mathcal{A}$ outputs $b' = 0$ if the pebbling configuration is *good* as per Definition 4.2, and $b' = 1$ otherwise.

4.1 The Circuit

We construct a family of circuits $\mathsf{G} := \{\mathsf{G}_d\}_{d \in \mathbb{N}}$ and show that the loss in security for G_d is sub-exponential in d. The circuit is designed keeping our high-level idea in mind. The circuit $\mathsf{G}_d := \mathsf{G}_d^0 \circ \mathsf{G}_d^\wedge \circ \mathsf{G}_d^\oplus$ consists of the three blocks $\mathsf{G}_d^\oplus$, $\mathsf{G}_d^\wedge$ and G_d^0, with underlying graphs denoted by $G_d^\oplus$, $G_d^\wedge$ and G_d^0, respectively. The graph $G_d^\oplus$ (see Figure 2.(b)) is a so-called *tower graph* [8], and is obtained from so-called pyramid graphs of depth d (see Figure 2.(a)).

– $\mathsf{G}_d^\oplus$ is obtained from $G_d^\oplus$ by substituting each vertex with an XOR gate as shown in Figure 2. On a high level, the pyramid structure ensures high pebbling complexity whereas the XOR gates preserve (most) entropy in the input, which makes it hard for a reduction to obtain correct evaluation of pebbled gates.
– G_d^0 consists of a binary tree of AND gates and its sole role is to set the output of the circuit G to constant 0.[11]
– $\mathsf{G}_d^\wedge$ sits in between the $\mathsf{G}_d^\oplus$ and G_d^0 blocks (see Figure 1), and consists of one AND gate serving as "control" gate for each XOR gate in $\mathsf{G}_d^\oplus$ and each input gate. Each AND gate g in $G_d^\wedge$ receives its inputs from (i) the output of its companion XOR gate in $G_d^\oplus$ (resp. input gate) and (ii) the XOR gate in the last layer of $G_d^\oplus$ in (vertical) alignment with g (see Figure 1, formal definition in the full version of this paper). As mentioned previously, intuitively, this block will act as an "error detection" mechanism for the $\mathsf{G}_d^\oplus$ block in the sense that it helps detect if (malformed) garblings of XOR gates evaluate wrongly.

For a precise description of the circuit and a proof that G is indeed constant, we refer to the appendix.

4.2 Vulnerability of the Circuit $\mathsf{G}^\oplus$

In Section 4.5 we will prove that any black-box reduction R that aims to use $\mathcal{A}$ to gain advantage in breaking the IND-CPA security of encryption scheme $\mathcal{F}$ has to simulate $(\tilde{\mathsf{G}}, \tilde{x})$ such that the extracted pebbling configuration on $\mathsf{G}^\oplus$ contains $d-1$ or d gray or black pebbles. Each of these pebbles implies that

[11] In principle we could have used constant-0 gates in place of the AND gates, or simply a single constant-0 gate of high fan-in (which would considerably simplify the description). But we prefer to stick to the standard Boolean basis.

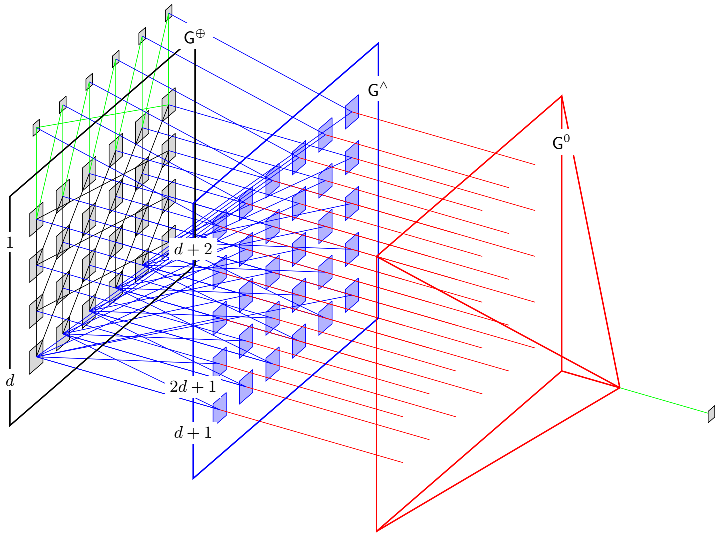


Fig. 1. Schematic diagram for the candidate circuit of width 5 and depth 4. The input and output wires are coloured green. The layer number is indicated on the left. The first two blocks are the XOR and AND layers respectively; the final pyramid denotes the binary tree.

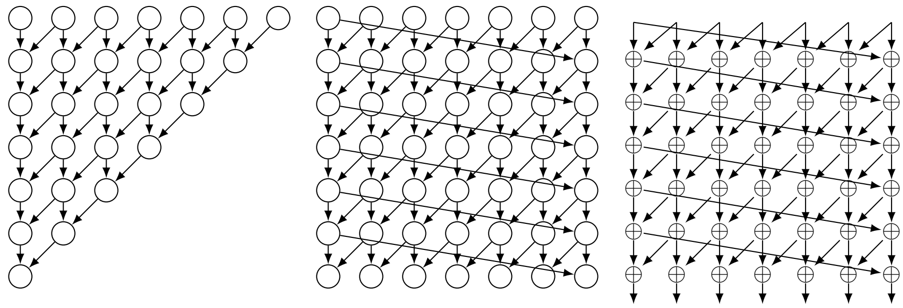
at least one of the ciphertexts associated to that gate must be malformed and modify the output of some input key pair. In the case that all AND gates are properly garbled, all keys can be mapped to bits and hence such a switch of the output can be detected (cf. Lemma 4.6). Thus, we consider the following game.

– On input a circuit C and a parameter d, R chooses a circuit C' of the same topology as C such that all except exactly d (non-input) gates coincide with the corresponding gates in C. R sends C' to $\mathcal{A}$.
– On receipt of C', $\mathcal{A}$ samples $x \leftarrow \{0,1\}^n$ uniformly at random.
– R wins if for all gates in C' the output during evaluation on input x coincides with the corresponding output bit when evaluating C.

We now prove that for $\mathsf{C} = \mathsf{G}^{\oplus}$, no algorithm R wins the above game with non-negligible (in d) probability.

Lemma 4.1. *Let $d \in [1,n]$. For $\mathsf{G} = \mathsf{G}^{\oplus}$ and any R, the probability that R wins the above game is at most $(\frac{3}{4})^{\sqrt{d}/4}$.*

First, note that all except d gates in G' are XOR gates, and in particular a linear function over $\mathbb{Z}_2$. For each of the remaining d malformed gates, on the other hand, at least one input pair is mapped to a different output bit than it would be in an XOR operation. We call the corresponding gates in the original circuit $\mathsf{G}^{\oplus}$ *pebbled*. To prove Lemma 4.1, we will show that there exists a subset of at least $\sqrt{d}/4$ of those d pebbled gates such that their input is determined by



(a) (b) (c)

Fig. 2. The graphs and the circuit for parameter $d = 6$: (a) A pyramid graph of depth d, (b) Extending the pyramid graph to get a tower graph $G_d^{\oplus}$ of depth d and (c) Circuit $\mathsf{G}_d^{\oplus}$ obtained replacing the vertices in $G_d^{\oplus}$ with XOR gates.

independent linear functions. This implies that instead of choosing $x \leftarrow \{0,1\}^n$, $\mathcal{A}$ can equivalently choose the $\sqrt{d}/2$ input bits uniformly at random, and then choose x uniformly under the constraint that the values running over these wires during evaluation of $\mathsf{G}^{\oplus}$ must be consistent with the predetermined bits. Clearly, x chosen this way is still uniformly random in $\{0,1\}^n$. By definition of the game, R only wins the game if for all gates in G' the output during evaluation on input x coincides with the corresponding output bit when evaluating G, and this must in particular also hold for the pebbled gates. Since each of the malformed gates in G' flips the output of at least one of the four possible input pairs, and the input bits of $\sqrt{d}/4$ of the pebbled gates were chosen independently and uniformly at random, the probability that R wins is at most $(\frac{3}{4})^{\sqrt{d}/4}$.

Towards proving Lemma 4.1, let M denote the linear mapping corresponding to one layer of gates in the circuit $\mathsf{G}^{\oplus}$, i.e., written in matrix notation,

$$M = \begin{pmatrix} 1\,1\,0 & \dots & 0\,0\,0 \\ 0\,1\,1 & \dots & 0\,0\,0 \\ \vdots & \ddots & \vdots \\ 0\,0\,0 & \dots & 0\,1\,1 \\ 1\,0\,0 & \dots & 0\,0\,1 \end{pmatrix}.$$

The output of the μth layer of $\mathsf{G}^{\oplus}$ on input $x \in \{0,1\}^n$ is given by $M^{\mu} \cdot x$, hence we denote the degree-1 polynomial in $\mathbb{Z}_2[x_1, \dots, x_n]$ which determines its ν-th bit by M_{ν}^{μ} (for $\mu \in [0,n]$ and $\nu \in [1,n]$). Denoting by $\overline{\nu+1}$ the representation of the residue class $\nu + 1 \mod n$ in $[n]$, we have e.g.,

$$M_\nu^0 = x_\nu, \quad M_\nu^1 = x_\nu \oplus x_{\overline{\nu+1}}, \quad M_\nu^2 = x_\nu \oplus x_{\overline{\nu+2}}, \quad M_\nu^3 = x_\nu \oplus x_{\overline{\nu+1}} \oplus x_{\overline{\nu+2}} \oplus x_{\overline{\nu+3}}$$

and in general it holds

$$M_\nu^\mu = M_\nu^{\mu-1} \oplus M_{\overline{\nu+1}}^{\mu-1} \tag{2}$$

for all $\mu \in \mathbb{N}, \nu \in [1, n]$. In the following we will associate gates with the corresponding polynomials that determine their outputs.

If the input length n is odd – for convenience we assume n to be one less than a power of 2 – then $\mathsf{G}^{\oplus}$ maintains high entropy; to prove this, we use the following explicit representation of the polynomials M^{μ}_{ν}.

Lemma 4.2 (explicit formula for the polynomials M^{μ}_{ν}). *Let $n = 2^{\kappa} - 1$, $\kappa \in \mathbb{N}$, M defined above, $\mu \in \mathbb{N}$, and $\nu \in [1, n]$. For $\overline{\mu} \neq n$ and $\beta_k \in \{0, 1\}$ its binary decomposition, i.e. $\overline{\mu} = \sum_{k \in [0,\kappa-1]} \beta_k 2^k$, it holds:*

$$M^{\mu}_{\nu} = \bigoplus_{i \in [1,n]} \alpha_i x_i, \text{ where } \alpha_i = \begin{cases} 1 & \text{if } i \in \nu + \sum_{k \in [0,\kappa-1]} \{0, \beta_k\} \cdot 2^k \mod n, \\ 0 & \text{else.} \end{cases} \tag{3}$$

Note, M^{μ}_{ν} only depends on $\overline{\mu}$, not on μ. For $\overline{\mu} = n = 2^{\kappa} - 1$, it holds:

$$M^{\mu}_{\nu} = \bigoplus_{i \in [1,n]} \alpha_i x_i, \text{ where } \alpha_i = \begin{cases} 1 & \text{if } i \neq \nu, \\ 0 & \text{else.} \end{cases} \tag{4}$$

A proof of Lemma 4.2 can be found in the full version of this paper. Lemma 4.2 directly implies several useful properties, which we summarize in the following corollary.

Corollary 4.1 (Properties of M and $\mathsf{G}^{\oplus}$). *For M defined as above, $n = 2^{\kappa} - 1$, $\kappa \in \mathbb{N}$, it holds*

1. *$M^{2^{\kappa}} = M$, which implies $\mathsf{rank}(M^k) = n - 1$ for all $k \geq 1$, i.e., $\mathsf{G}^{\oplus} = M^d$ is 2-to-1 for any d.*
2. *Any $n - 1$ output bits of M^k ($k \geq 1$) are determined by linearly independent degree-1 polynomials.*
3. *$\mathsf{Image}(\mathsf{G}^{\oplus}) = \{x = (x_1, \ldots, x_n) \in \{0, 1\}^n \mid \bigoplus_{i \in [1,n]} x_i = 0\}$, i.e., all vectors in the image of $\mathsf{G}^{\oplus}$ contain an even number of 1s.*

The first property immediately follows from Lemma 4.2 since for $\mu = 2^{\kappa}$ we have $\overline{\mu} = 1$. The second property then follows from $\mathsf{rank}(M^k) = n - 1$. For the last property, note that the set $\nu + \sum_{k \in [0,\kappa-1]} \{0, \beta_k\} \cdot 2^k \mod n$ is even whenever a single bit β_k is nonzero (which is true for all $\overline{\mu} > 0$), and also the set $\{i \in [n] \mid i \neq \nu\}$ is even since n is odd.

The following Lemma immediately implies Lemma 4.1, a proof can be found in the full version of this paper.

Lemma 4.3. *Any subset $\mathcal{S} \subset \{M^{\mu}_{\nu}\}_{\mu \in [0,n], \nu \in [1,n]}$ of polynomials in $\mathbb{Z}_2[x_1, \ldots, x_n]$ with $s := |\mathcal{S}|$ contains a subset $\mathcal{S}'$ of size $\sqrt{s}/4$ such that $|\mathsf{parents}(\mathcal{S}')| = \sqrt{s}/2$ and $\mathsf{parents}(\mathcal{S}')$ is linearly independent, where $\mathsf{parents}(M^{\mu}_{\nu}) := \{M^{\mu-1}_{\nu}, M^{\mu-1}_{\nu+1}\}$.*

Lemma 4.1 now follows, since for any set of d pebbled gates, by Lemma 4.3 there exists a subset $\mathcal{S}'$ of $\sqrt{d}/4$ pebbled gates such that their parents are distinct and form a linearly independent set.

4.3 Pebbling Game and Threshold

Recall that in Yao's garbling scheme, each gate g is associated with a (honest) garbling table $\tilde{\boldsymbol{g}}$, which consists of four double encryptions that encode g's gate table. However, a reduction is free to alter the contents of the honest garbling table in any way. In fact, the upper bounds in [16,14] crucially rely on the ability to do this in an indistinguishable manner: in the real game the garbling tables are all honest, whereas in the simulated game the garbling tables all encode the constant-0 gate, and the hybrids involve replacing the honest garbling tables one by one with that of the constant-0 gate.[12] We introduce a pebble game to precisely model such different simulations of the garbled circuit $\tilde{\mathsf{G}}$ (by the reduction). Loosely speaking, the extracted pebble configuration is an abstract representation of the simulation $(\tilde{\mathsf{G}}, \tilde{x}_b)$, and the pebbling rules model the reduction's ability to maul garbling tables in $\tilde{\mathsf{G}}$ without being noticed (indistinguishability).

The pebbles. Intuitively, the pebble on a gate g encodes how "different" the garbling table $\tilde{\boldsymbol{g}}'$ which $\mathcal{A}$ receives is from an honest garbling $\tilde{\boldsymbol{g}}$. To this end, we employ three different pebbles: white, gray and black.

- A white pebble on g indicates that $\tilde{\boldsymbol{g}}'$ and $\tilde{\boldsymbol{g}}$ are at "distance" 0 (defined below), i.e., $\tilde{\boldsymbol{g}}$ is (distributed identically to) an honest garbling table of g.
- A gray or black pebble on g indicates that $\tilde{\boldsymbol{g}}'$ is malformed. What differentiates gray from black is the degree of malformation: loosely speaking, a gray pebble indicates that $\tilde{\boldsymbol{g}}'$ is at a distance 1 from $\tilde{\boldsymbol{g}}$, whereas a black pebble indicates that $\tilde{\boldsymbol{g}}'$ is at a distance 2 (or more).

To understand what we mean by distance, we need to take a closer look at the structure of a garbling table. An honest garbling table $\tilde{\boldsymbol{g}}$ consists of the four double encryptions shown in Table 1.(a). We assign a gray pebble to a gate g if the garbling table of g in $\tilde{\mathsf{G}}$ can be proven indistinguishable from $\tilde{\boldsymbol{g}}$ by embedding a *single* IND-CPA challenge key (among k_u^0, k_u^1, k_v^0 and k_v^1). For example, let's consider an AND gate and its honest garbling table (Table 1.(b)): a malformed table that is at distance one (via the key k_u^1 or k_v^1) from it is, e.g., a garbling table that encodes the constant-0 gate (Table 1.(d)). A garbling of an XOR gate, in contrast, is at distance 2 from a garbling of a constant gate: If k_u^a and k_v^b are the keys revealed during evaluation, then the garbling of an XOR gate can be proven indistinguishable from the constant-$(a \oplus b)$ gate only by first embedding a challenge key at k_u^{1-a} and then a second challenge key at k_v^{1-b}, or vice versa; i.e. the reduction needs to embed challenges at each input wire.

[12] Note, this simulation crucially relies on the fact that keys can be *equivocated*: While the output keys are all associated to 0, when altering the output mapping accordingly evaluation will still succeed. Note that in the selective setting for Yao's original scheme as well as in the adaptive setting for the modified scheme [14] the input is known before the output mapping is sent.

$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^0}(k_w^{g(0,0)}))$	$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^0}(k_w^0))$	$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^0}(k_w^0))$	$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^0}(k_w^0))$
$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^0}(k_w^{g(1,0)}))$	$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^0}(k_w^0))$	$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^0}(k_w^1))$	$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^0}(k_w^0))$
$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^1}(k_w^{g(0,1)}))$	$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^1}(k_w^0))$	$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^1}(k_w^1))$	$\mathsf{E}_{k_u^0}(\mathsf{E}_{k_v^1}(k_w^0))$
$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^1}(k_w^{g(1,1)}))$	$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^1}(k_w^1))$	$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^1}(k_w^0))$	$\mathsf{E}_{k_u^1}(\mathsf{E}_{k_v^1}(k_w^0))$
(a)	(b)	(c)	(d)

Table 1. Garbling tables for (a) general gate g, (b) AND gate, (c) XOR gate, and (d) constant-0 gate. u and v denote the two input wires, whereas w denotes the output wire.

Pebbling rules. To complete the description of a pebble game, we need to describe the pebbling rules. These rules essentially capture the following observation: a reduction (with overwhelming probability) cannot possess encryptions of its (IND-CPA) challenge key. Therefore, whenever the garbling table $\tilde{\boldsymbol{g}}$ of a gate g has been switched to a malformed garbling $\tilde{\boldsymbol{g}}'$ (say) at distance one, (at least) one of the garbling tables associated to its predecessor gates, say g_u, *must* have been first switched to a garbling that encodes only one of g_u's output keys. This is required to "free up" one of g_u's output keys (so that it can now be set as the challenge key). Looking ahead, we will be interested in pebbling the circuit $\mathsf{G}^{\oplus}$ which consists of XOR gates only. Hence, the pebbling rules are designed to capture the structure of XOR gates. Recall that an XOR gate is at distance 2 from a constant gate, thus, we end up with the following rules (where g_u and g_v denote the two predecessors of g):

1. a gray pebble can be placed on or removed from a gate g only if (at least) one of its predecessor gates (say g_u) carries a black pebble; and
2. a gray pebble on a gate g can be swapped with a black pebble if the *other* predecessor gate (i.e., g_v) carries a black pebble.

The actual game. The above white-gray-black (`WGB`) pebble game is a simplified version of the (`WG`^{3}`B`) pebble game we end up using, but it is sufficient to convey the essential ideas that we use. The actual game, defined in Definition 4.1 (Section 4.3), is more fine-grained: in order to keep track of the inner and outer encryptions, we introduce three types of gray pebbles (gray-left, gray-right and gray-free), and the pebbling rules are also modified accordingly.

Definition 4.1 (Reversible `WG`^{3}`B` pebbling game for indegree-2 graphs). *Consider a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = [1, S]$ and let $\mathcal{X} = \{\mathtt{W}, \mathtt{G}_*, \mathtt{G}_L, \mathtt{G}_R, \mathtt{B}\}$ denote the set of colours of the pebbles. Consider a sequence $\boldsymbol{\mathcal{P}} := (\mathcal{P}_0, \ldots, \mathcal{P}_\tau)$ of pebbling configurations for G, where $\mathcal{P}_i \in \mathcal{X}^{\mathcal{V}}$ for all $i \in [0, \tau]$. We call such a sequence* a `WG`^{3}`B` pebbling strategy *for G if the following two criteria are satisfied:*

1. *In the initial configuration all the vertices are pebbled white (i.e., $\mathcal{P}_0 = (\mathtt{W}, \ldots, \mathtt{W})$) and in the final configuration* at least one sink *of $\mathtt{G}$ is pebbled gray (i.e., $\mathcal{P}_\tau = (\ldots, \mathtt{G}_., \ldots)$), where $\mathtt{G}_.$ denotes an arbitrary type of gray, i.e. $\mathtt{G}_. \in \{\mathtt{G}_*, \mathtt{G}_L, \mathtt{G}_R\}$.*

2. *Two subsequent configurations differ only in one vertex and the following rules are respected in each move:*
   (a) $\mathtt{W} \leftrightarrow \mathtt{G}_*$*: a white pebble can be replaced by a* $\mathtt{G}_*$ *pebble (and vice versa) if one of its parents is black-pebbled*
   (b) $\mathtt{W}/\mathtt{G}_* \leftrightarrow \mathtt{G}_L$*: a white or* $\mathtt{G}_*$ *pebble can be replaced by a* $\mathtt{G}_L$ *pebble (and vice versa) if its* left *parent is black-pebbled*
   (c) $\mathtt{W}/\mathtt{G}_* \leftrightarrow \mathtt{G}_R$*: a white or* $\mathtt{G}_*$ *pebble can be replaced by a* $\mathtt{G}_R$ *pebble (and vice versa) if its* right *parent is black-pebbled*
   (d) $\mathtt{G}_L \leftrightarrow \mathtt{B}$*: a* $\mathtt{G}_L$ *pebble can be replaced by a black pebble (and vice versa) if its* right *parent is black-pebbled*
   (e) $\mathtt{G}_R \leftrightarrow \mathtt{B}$*: a* $\mathtt{G}_R$ *pebble can be replaced by a black pebble (and vice versa) if its* left *parent is black-pebbled*

The space-complexity *of a* $\mathtt{WG^3B}$ *pebbling strategy* $\boldsymbol{\mathcal{P}} = (\mathcal{P}_0, \ldots, \mathcal{P}_\tau)$ *for a DAG* G *is defined as*

$$\sigma_G(\boldsymbol{\mathcal{P}}) := \max_{i \in [0,\tau]} |\{j \in [1,S] : \mathcal{P}_i(j) \in \{\mathtt{G}_*, \mathtt{G}_L, \mathtt{G}_R, \mathtt{B}\}\}|.$$

For a subgraph G' *induced on vertex set* $\mathcal{V}' \subset \mathcal{V}$*, the space-complexity of* $\boldsymbol{\mathcal{P}}$ *restricted to* G' *is defined as*

$$\sigma_{|G'}(\boldsymbol{\mathcal{P}}) := \max_{i \in [0,\tau]} |\{j \in \mathcal{V}' : \mathcal{P}_i(j) \in \{\mathtt{G}_*, \mathtt{G}_L, \mathtt{G}_R, \mathtt{B}\}\}|.$$

The space-complexity of a DAG G *is the minimum space-complexity over all of its strategies* $\boldsymbol{\mathcal{P}}^G$*:*

$$\sigma(G) := \min_{\boldsymbol{\mathcal{P}} \in \boldsymbol{\mathcal{P}}^G} \sigma_G(\boldsymbol{\mathcal{P}}). \tag{5}$$

The following lemma gives a lower bound on the $\mathtt{WG^3B}$ pebbling complexity of the graph $G \setminus G^0$ underlying the first two blocks $\mathsf{G}^\wedge \circ \mathsf{G}^\oplus$ of our candidate circuit G. A proof can be found in the full version of this paper.

Lemma 4.4 (Pebbling lower bound on $G \setminus G^0$). *Let* $G \setminus G^0$ *be the graph underlying the circuit* $\mathsf{G}^\wedge \circ \mathsf{G}^\oplus$*. To gray-pebble a gate on layer* $d' \in [1, d+1]$ *following the reversible* $\mathtt{WG^3B}$ *pebbling rules from Definition 4.1, one requires space-complexity at least* $d' - 1$*. Furthermore, to* $\mathtt{G}_L$*- or* $\mathtt{B}$*-pebble a gate on layer* $d' \geq d+1$*, one requires at least* d *gray or black pebbles simultaneously on the first* d *layers.*

The following definition now gives a *cut in the configuration graph*; our adversary $\mathcal{A}$ will be a *threshold* adversary with respect to this cut.

Definition 4.2 (Good pebbling configurations). *A pebbling configuration* $\mathcal{P}$ *on DAG* $G \setminus G^0$ *is called* good *if it is* reachable *by reversible* $\mathtt{WG^3B}$ *pebbling moves using less than* d *gray or black pebbles on the first* d *layers simultaneously, i.e., there exists a* $\mathtt{WG^3B}$ *pebbling strategy* $\boldsymbol{\mathcal{P}} := (\mathcal{P}_0, \ldots, \mathcal{P})$ *for* G *such that* $\sigma_{|G^\oplus}(\boldsymbol{\mathcal{P}}) \leq d - 1$*.*

In particular, by Lemma 4.4, any pebbling configuration $\mathcal{P}$ with a $\mathtt{G}_L$ or $\mathtt{B}$ pebble on a gate in $\mathsf{G}^\wedge$ is bad.

4.4 Extraction of Pebbling Configuration on $G \setminus G^0$

In this section we will discuss how to extract such a pebbling configuration. Note, that $\mathcal{A}$ is computationally unbounded, hence can extract messages and keys from ciphertexts by brute-force search.

1. First, check whether $(\tilde{\mathsf{G}}, \tilde{x})$ *evaluates correctly*, i.e., $\mathsf{GEval}(\tilde{\mathsf{G}}, \tilde{x}) = \mathsf{G}(x_0)$.
   If the evaluation check passes, check whether $\tilde{\mathsf{G}}, \tilde{x}$ have the *correct syntax*: Check whether $\tilde{\mathsf{G}}$ consists of four ciphertexts for each gate, which have the following form
   $$c_1 = \mathsf{Enc}_{k_1}(\mathsf{Enc}_{k_3}(k_5)),\ c_2 = \mathsf{Enc}_{k_1}(\mathsf{Enc}_{k_4}(m_2)),\\ \{c_3, c_4\} = \{\mathsf{Enc}_{k_2}(m_3), \mathsf{Enc}_{k_2}(m_4)\}, \tag{6}$$
   for distinct keys k_1, k_2, k_3, k_4, k_5 and arbitrary (not necessarily distinct) messages m_2, m_3, m_4, where keys k_1 and k_3 are revealed during evaluation $\mathsf{GEval}(\tilde{\mathsf{G}}, \tilde{x})$. I.e., two of the four ciphertexts are encryptions under the same left secret keys k_1 and k_2, respectively, one of them is a double encryption $\mathsf{Enc}_{k_1}(\mathsf{Enc}_{k_3}(k_5))$ under left key k_1 and some right key k_3 of an output key k_5 (all these being revealed throughout evaluation), and the second encryption under k_1 encrypts an encryption under a second right key k_4 (of an arbitrary message m_2).
   Finally, check *consistency of keys*: For each gate, extract key pairs (k_1, k_2) and (k_3, k_4) corresponding to left and right input wires, and check whether they are consistent with the keys extracted from sibling gates: If gate g is the left sibling of g', then g's right input key pair must coincide with the left key pair extracted from g', i.e., $(k_3, k_4) = (k_1', k_2')$. Note, if this check passes, then all wires in the circuit can be uniquely associated with a key pair. Finally, check that all extracted keys are distinct.
   If any of these checks fails, map $(\tilde{\mathsf{G}}, \tilde{x})$ to a bad pebbling configuration, e.g., to the pebbling configuration on $\mathtt{G}$ where all gates at levels $[d+1, 2d+1]$ are black pebbled[13] and quit.

Remark 4.1. Note, syntax and consistency checks allow a reduction to distinguish

– a ciphertext from a non-ciphertext,
– a ciphertext under key k from a ciphertext under key $k' \neq k$.

We will argue in Section 4.5 that this is of no help to the reduction for breaking IND-CPA security of the information-theoretic encryption scheme $\mathcal{F}$.

For all garblings $(\tilde{\mathsf{G}}, \tilde{x})$ that pass correctness, syntax, and consistency checks, $\mathcal{A}$ will extract a pebbling configuration on $G \setminus G^0$ by mapping each gate to a color in $\{\mathtt{W}, \mathtt{G}_*, \mathtt{G}_L, \mathtt{G}_R, \mathtt{B}\}$.

[13] This choice was made for convenience (see Lemmas 4.6 to 4.8), but in principle could be an arbitrary bad configuration, and should simply guarantee that no reduction can gain any advantage by departing from the protocol in an obvious way.

2. For each XOR gate g_j ($j \in [1, d] \cdot n + [0, n]$): Check whether g_j is garbled correctly with respect to input x_0. To this aim, let b_l, b_r, and $b_o = g_j(b_l, b_r) = b_l \oplus b_r$ denote the left/right input and the output bit of g_j, respectively, when evaluating G on x_0. We use the same notation as in Equation 6 above; furthermore, let k_6 be the second key associated with the output wire (which was extracted from the garbling tables of the successor gates).
   - If g_j is garbled similar to the case of an honest garbling of (G, x_0), i.e., $m_2 = k_6$, $m_3 = \mathsf{Enc}_{k_3}(k_6)$, and $m_4 = \mathsf{Enc}_{k_4}(k_5)$ (or the roles of m_3, m_4 permuted), then associate g_j with a $\mathtt{W}$ pebble.
   - If m_2 and m_3 are as in the previous case, but $m_4 = \mathsf{Enc}_{k_4}(m)$ for some message $m \neq k_5$, then associate g_j with a $\mathtt{G}_*$ pebble. Similarly for the case where the roles of m_3, m_4 are permuted.
   - If m_3 is as in the first case, $m_4 = \mathsf{Enc}_{k_4}(m)$ for an arbitrary message m, but $m_2 \neq k_6$, then associate g_j with a $\mathtt{G}_R$ pebble. Similarly for the case where the roles of m_3, m_4 are permuted.
   - If $m_2 = k_6$ is as in the first case, but $\{m_3, m_4\}$ differs from the previous cases, then associate g_j with a $\mathtt{G}_L$ pebble.
   - For all other cases, associate g_j with a $\mathtt{B}$ pebble.

Remark 4.2. Due to symmetry of the XOR operation, whether a gate is considered properly garbled (i.e. mapped to a white pebble) or not (i.e. mapped to gray or black) does *not* depend on the input keys. Thus, the set of black and gray pebbles on $G^{\oplus}$ can be extracted *independently* of x_0 and $\tilde{x}$.

3. For each AND gate g_j ($j \in [d+1, 2d+1] \cdot n + [0, n]$): Similar to the case of XOR gates, check whether the gate is correctly garbled with respect to x_0. Using the same notation as above, associate g_j with a pebble as follows:
   - If g_j is garbled similar to the case of an honest garbling of (G, x_0), i.e., for
     $(b_l, b_r) = (0, 0)$, we have $m_2 = k_5$, $m_3 = \mathsf{Enc}_{k_3}(k_5)$, and $m_4 = \mathsf{Enc}_{k_4}(k_6)$,
     $(b_l, b_r) = (0, 1)$, we have $m_2 = k_5$, $m_3 = \mathsf{Enc}_{k_3}(k_6)$, and $m_4 = \mathsf{Enc}_{k_4}(k_5)$,
     $(b_l, b_r) = (1, 0)$, we have $m_2 = k_6$, $m_3 = \mathsf{Enc}_{k_3}(k_5)$, and $m_4 = \mathsf{Enc}_{k_4}(k_5)$,
     $(b_l, b_r) = (1, 1)$, we have $m_2 = k_6$, $m_3 = \mathsf{Enc}_{k_3}(k_6)$, and $m_4 = \mathsf{Enc}_{k_4}(k_6)$,
     (or the roles of m_3, m_4 permuted) then associate g_j with a $\mathtt{W}$ pebble.
   - If m_2 and m_3 are as in the previous case, but $m_4 = \mathsf{Enc}_{k_4}(m)$ for some message m that differs from above, then associate g_j with a $\mathtt{G}_*$ pebble. (Similarly for the case where the roles of m_3, m_4 are permuted.)
   - If m_3 is as in the first case, $m_4 = \mathsf{Enc}_{k_4}(m)$ for an arbitrary message m, but m_2 differs from the previous case, then associate g_j with a $\mathtt{G}_R$ pebble. (Similarly for the case where the roles of m_3, m_4 are permuted.)
   - If m_2 is as in the first case, but $\{m_3, m_4\}$ differs from the previous cases, then associate g_j with a $\mathtt{G}_L$ pebble.
   - For all other cases, associate g_j with a B pebble.

Remark 4.3. At first sight, it might seem counterintuitive that the mapping from gates to colours not only depends on the associated ciphertexts, but also on the input x_0. This however is unavoidable since the adversary $\mathcal{A}$ cannot simply map keys to bits, but can only *relate* them to the keys it learned from $\tilde{x}$, which might be properly garbled or not.

In the following lemma, we prove that the adversary $\mathcal{A}$ using the above pebbling extraction indeed breaks indistinguishability of Yao's garbling scheme. A proof can be found in the full version of this paper.

Lemma 4.5. *$\mathcal{A}$ breaks indistinguishability of the garbling scheme with probability $1 - 1/2^{n-1}$.*

Since $\mathcal{A}$ extracts the pebble mode of a gate with regard to the garbled input (i.e., the keys it learns through evaluation), the reduction can still change the mode of a gate *after* it output $\tilde{\mathtt{G}}$ by choosing different input keys for $\tilde{x}$. In the following lemmas we prove that this flexibility of choosing the input keys is of not much help to a reduction aiming at a good pebbling configuration, where in particular all gates at layers $[d+1, 2d+1]$ are mapped to $\mathtt{W}$, $\mathtt{G}_*$, or $\mathtt{G}_R$ pebbles.

First, we consider the case of a properly garbled AND gates. In this case, due to the asymmetry of the AND operation, input keys can be associated with bits and hence a properly garbled layer of AND gates has a similar function as an output mapping. A proof can be found in the full version of this paper.

Lemma 4.6. *For any garbling of an AND gate on layer $[d+1, 2d+1]$, and any input bits b_l, b_r, there exists at most one input key pair (k_1, k_3) such that the gate will be mapped to a $\mathtt{W}$ pebble.*

The situation becomes a bit more involved if AND gates are not properly garbled, since in this case asymmetry might be broken. However, if the left input keys can be mapped to bits, then we can still obtain some meaningful guarantees. We first consider the case that an AND gate is garbled in $\mathtt{G}_*$ mode, i.e. one ciphertext is malformed and there exist some input bits (b_l, b_r) such that it will be mapped to a $\mathtt{G}_*$ pebble. In the following Lemma we prove that for a different right input bit $1 - b_r$ the gate will be mapped to a $\mathtt{G}_L$ pebble instead. A proof can be found in the full version of this paper.

Lemma 4.7. *For any garbling of an AND gate, any left input bit b_l, and fixed left input key, there exists at most one $b_r \in \{0,1\}$ such that there exists a (not necessarily unique) right input key such that the gate will be mapped to a $\mathtt{G}_*$ pebble. If such a right input bit b_r exists, then for right input bit $1 - b_r$ the gate will be mapped to a $\mathtt{G}_L$ pebble.*

Next we consider the case of an AND gate that is garbled in $\mathtt{G}_R$ mode w.r.t. some input bits (b_l, b_r). In this case we have to distinguish two different ways to garble a gate such that it will be mapped to a $\mathtt{G}_R$ pebble. For one type of $\mathtt{G}_R$ pebble we can map keys to bits, just as in the case of properly garbled gates. For the second type of $\mathtt{G}_R$ pebble we obtain a similar guarantee as for $\mathtt{G}_*$ pebbles. A proof can be found in the full version of this paper.

Lemma 4.8. *For any garbling of an AND gate on layer $[d+1, 2d+1]$, any left input bit b_l, and fixed left input key, one of the following is true:*

1. *For any right input bit $b_r \in \{0,1\}$ there exists at most one right input key such that the gate will be mapped to a $\mathtt{G}_R$ pebble. If such a key exists, then for any other right input key the gate will be mapped to a $\mathtt{B}$ pebble.*
2. *There exists at most one input bit $b_r \in \{0,1\}$ such that there exists a right input key k_r such that the gate will be mapped to a $\mathtt{G}_R$ pebble. If such a bit exists, then for right input bit $1 - b_r$ and any right input key the gate will be mapped to a $\mathtt{B}$ pebble.*

These two cases characterize two different types of $\mathtt{G}_R$ pebbled gates, where we denote a gate as $\mathtt{G}_R$-type-1 if case 1 is true, and $\mathtt{G}_R$-type-2 if only *case 2 is true.*

4.5 Lower Bound on Security Loss for any Reduction

In this section we will combine all previous results to prove a lower bound on adaptive security of Yao's garbling scheme. More precisely, we will prove that any black-box reduction which aims to exploit $\mathcal{A}$'s distinguishing advantage to break IND-CPA security of the underlying encryption scheme loses a factor subexponential in the depth of the circuit.

Let R be an arbitrary PPT reduction which has black-box access to an adversary $\mathcal{A}$ that breaks indistinguishability of Yao's garbling scheme, and attempts to solve an IND-CPA challenge with respect to an encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Following the approach of Kamath et al. [15], we define an *information-theoretically* secure encryption scheme $\mathcal{F} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows: For $l \in \{1, 6\}$, let $\mathsf{E}_l : \{0,1\}^{(l+2)\lambda} \to \{0,1\}^{2(l+2)\lambda}$ be a random expanding function (which is injective with overwhelming probability).

- Key generation $\mathsf{Gen}(1^\lambda)$: On input a security parameter λ in unary, output a key $k \leftarrow \{0,1\}^*$ uniformly at random.
- Encryption $\mathsf{Enc}(k, m)$: On input a key $k \in \{0,1\}^\lambda$ and a message $m \in \{0,1\}^{l \cdot \lambda}$ with $l \in \{1, 6\}$, sample randomness $r \leftarrow \{0,1\}^\lambda$, and output $\mathsf{E}_l(k, m; r)$.
- Decryption $\mathsf{Dec}(k, c)$ is simulated to be consistent with Enc: On input a key $k \in \{0,1\}^\lambda$ and a ciphertext $c \in \{0,1\}^{2(l+2)\lambda}$ with $l \in \{1, 6\}$, check whether c lies in the image of $\mathsf{E}_l(k, \cdot; \cdot)$, if so extract $m \in \{0,1\}^{l \cdot \lambda}$, $r \in \{0,1\}^\lambda$ such that $c = \mathsf{E}_l(k, m; r)$ and output m, otherwise output $\perp$.

Choosing E_l ($l \in \{1, 6\}$) to be random functions implies that $\mathcal{F}$ is information-theoretically IND-CCA secure. Thus, since R only makes polynomially many queries, the only non-negligible advantage R has in breaking the IND-CPA security of $\mathcal{F}$ must stem from its interaction with $\mathcal{A}$. Furthermore, with all but negligible (in λ) probability $\mathcal{F}$ satisfies the special property (Definition 3.3), hence can be used in Yao's garbling scheme.

We first argue that neither checking correctness, syntax, nor consistency (cf. Section 4.4) is of any help to R. Obviously, this is true for the correctness check, since R can efficiently evaluate $\mathsf{GEval}(\tilde{\mathsf{G}}, \tilde{x})$. However, we have to argue a bit more to prove that also syntax and consistency checks are of no help to R. To this aim, we construct an oracle $\mathcal{O}$ that allows to distinguish

- a ciphertext from an arbitrary string in $\{0,1\}^{2(l+2)\lambda}$ for $l \in \{1,6\}$,
- a ciphertext under key $k \in \{0,1\}^{\lambda}$ from a ciphertext under key $k' \neq k$.

More precisely, $\mathcal{O}$ takes as input two strings $s \in \{0,1\}^{2(l+2)\lambda}$ and $s' \in \{0,1\}^{2(l'+2)\lambda}$ ($l, l' \in \{1,6\}$) and checks whether s, s' lie in the image of $\mathsf{E}_l, \mathsf{E}_{l'}$, respectively. If this check fails for one of the strings, then $\mathcal{O}$ outputs $\perp$. Otherwise, it extracts preimages $(k, m, r) \in \{0,1\}^{(l+2)\lambda}$ under E_l and $(k', m', r') \in \{0,1\}^{(l'+2)\lambda}$ under $\mathsf{E}_{l'}$. If $k = k'$, $\mathcal{O}$ outputs 1, otherwise 0.

In the full version of this paper we first show that access to oracle $\mathcal{O}$ allows R to efficiently carry out syntax and consistency checks, and then prove that $\mathcal{F}$ remains information-theoretically IND-CPA secure even against adversaries that have access to $\mathcal{O}$.

Now, to prove that any black-box reduction from indistinguishability of Yao's garbling scheme to IND-CPA security of the underlying encryption scheme suffers from a loss that is subexponential in the depth δ of the circuit, we construct an adversary $\mathcal{A}[c^*]$ that behaves just like $\mathcal{A}$ but doesn't decrypt challenge ciphertext c^*. More precisely, $\mathcal{A}[c^*]$ with input a ciphertext c^*, has oracle access to $\mathcal{O}$, $\mathcal{F}$, as well as an IND-CCA decryption oracle Dec_{k^*} that it can query on any ciphertext $c \neq c^*$. We construct $\mathcal{A}[c^*]$ such that it never decrypts c^* unless it already knows the encryption key k^* from other keys and ciphertexts in $\tilde{\mathsf{G}}, \tilde{x}$:

- First $\mathcal{A}[c^*]$ runs evaluation, syntax, and consistency checks using oracle $\mathcal{O}$. If these checks pass, similar to $\mathcal{A}$, the algorithm $\mathcal{A}[c^*]$ uses brute-force search to decrypt all ciphertexts *except* for those encrypted under k^* (to check whether a ciphertext is encrypted under k^* it uses $\mathcal{O}$ and c^*). Ciphertexts $c \neq c^*$ encrypted under k^* it decrypts using oracle Dec_{k^*}. For c^*, there are two cases:
    - If the key k^* was learned from previous decryptions (this can be checked by decrypting c^* under all known keys), $\mathcal{A}[c^*]$ simply decrypts c^* using k^*.
    - If the k^* is not known to $\mathcal{A}[c^*]$, then it simply *assumes* $c^* \in \{0,1\}^{2(l+2)\lambda}$ with $l \in \{1,6\}$ would decrypt to $0^{l \cdot \lambda}$.

  $\mathcal{A}[c^*]$ then continues analogous to $\mathcal{A}$ by mapping $(\tilde{\mathsf{G}}, \tilde{x})$ to a pebbling configuration and outputting 0 whenever the pebbling configuration is good per Definition 4.2, and 1 otherwise.

Clearly, since $\mathcal{A}[c^*]$ never decrypts c^* except if k^* is known, there is no chance for R to use $\mathcal{A}[c^*]$ to break IND-CPA security of $\mathcal{F}$.[14] It remains to bound the success probability of any PPT distinguisher D to distinguish $\mathcal{A}[c^*]$ from $\mathcal{A}$.[15] To this aim, we will first show how the $\mathtt{WG}^3\mathtt{B}$ pebbling game relates to this issue. A proof of the following Lemma can be found in the full version of this paper.

[14] Recall that our ideal encryption scheme $\mathcal{F}$ is IND-CCA secure, hence access to the oracle Dec_{k^*} used by $\mathcal{A}[c^*]$ is of no help to R.

[15] Note, we assume that $\mathcal{A}[c^*]$ has *private* access to its oracles and D cannot observe its oracle queries to distinguish it from $\mathcal{A}$.

Lemma 4.9. *Let $c^* \leftarrow \mathsf{Enc}_{k^*}(m)$ be an arbitrary ciphertext and let $\mathcal{P}$, $\mathcal{P}^*$ be the two pebbling configurations extracted by $\mathcal{A}$ and $\mathcal{A}[c^*]$, respectively, in the same execution of the game, i.e. using the same randomness. Then $\mathcal{P}^*$ differs from $\mathcal{P}$ by at most one valid $\mathtt{WG^3B}$ pebbling move.*

We will now bound the distinguishing advantage of $\mathsf{D}^{\mathcal{F}}$. Recall that a pebbling configuration on $G \setminus G^0$ is good per Definition 4.2 if it can be reached by $\mathtt{WG^3B}$ pebbling moves using at most $d-1$ pebbles on the first d layers. Thus, by Lemma 4.9, any successful distinguisher D has to simulate $\tilde{\mathsf{G}}$ and $\tilde{x}$ such that the pebbling configurations $\mathcal{P}, \mathcal{P}^*$ on G extracted by $\mathcal{A}$ and $\mathcal{A}[c^*]$, respectively, contain exactly $d-1$ or d black and gray pebbles on the first d layers (depending on the IND-CPA challenge bit b^*), contain only $\mathtt{W}$, $\mathtt{G}_*$, and $\mathtt{G}_R$ pebbles on higher layers, and differ by a valid $\mathtt{WG^3B}$ pebbling move within layers $[1, d+1]$.

In the following we will first restrict our analysis to *non-rewinding* distinguishers and assume x_0, x_1 were chosen uniformly at random by $\mathcal{A}$ *after* it sees $\tilde{\mathsf{G}}$. Finally we will discuss how to slightly modify our adversary $\mathcal{A}$ to also cover the case that D chooses $\mathcal{A}$'s randomness and rewinds $\mathcal{A}$.

To bound the success probability of D, let r be arbitrary random coins and consider two cases:

(1) there exists s such that the output of $\mathcal{A}(s)$ and $\mathcal{A}[c^*](s)$ after interaction with $\mathsf{D}(r, c^*)$ differs and in $\mathcal{P}$ and $\mathcal{P}^*$ there are *more than* $\bar{d}$ $\mathtt{G}_*$ and $\mathtt{G}_R$-type-2 (as defined in Lemma 4.8) pebbles in layers $[d+2, 2d+1]$,

(2) there exists s such that the output of $\mathcal{A}(s)$ and $\mathcal{A}[c^*](s)$ after interaction with $\mathsf{D}(r, c^*)$ differs and in $\mathcal{P}$ and $\mathcal{P}^*$ there are *at most* $\bar{d}$ $\mathtt{G}_*$ and $\mathtt{G}_R$-type-2 pebbles in layers $[d+2, 2d+1]$.

We leave the parameter $\bar{d} < d/3$ undefined for now and optimze it later. In Lemmas 4.10 and 4.11, we will argue that, intuitively, in both cases the distinguisher D must have correctly guessed many of the input bits in x_0.

Lemma 4.10. *Let r be arbitrary coins such that case (1) is true. Then the probability (over uniformly random coins s) that the output of $\mathcal{A}(s)$ and $\mathcal{A}[c^*](s)$ differs after interaction with $\mathsf{D}(r, c^*)$ is at most $(3/4)^{\sqrt{\bar{d}}/7}$.*

Proof. To prove this lemma, we will use Lemmas 4.6 to 4.8. First, note that D can only succeed if at most one of the gates at layer $d+1$ is not mapped to a $\mathtt{W}$ pebble, since the adversary $\mathcal{A}$ outputs 1 whenever any gate at layer $d+1$ is not $\mathtt{W}$ pebbled. Now, by Lemma 4.6, there is at most one pair of input keys to an AND gate that leads to this gate being mapped to a $\mathtt{W}$ pebble. As the input to all but one gate at layer $d+1$ comprises *all* input to layer $d+1$, this implies that D can only succeed, if it properly garbles all gates at layer $d+1$ and the input keys which are revealed through $\mathsf{GEval}(\tilde{\mathsf{G}}, \tilde{x})$ are associated with the corresponding bits in $\mathsf{G}^{\oplus}(x_0)$.

Next, consider the AND gates at layers $[d+2, 2d+1]$. For D to succeed, these gates must *not* end up $\mathtt{G}_L$ or $\mathtt{B}$ pebbled. Since all these gates have their left input from layer d and by the previous argument all these keys are fixed, we can apply

Lemmas 4.7 and 4.8: Let $\mathcal{S}$ denote the set of $\bar{d}$ gates in layers $[d+2, 2d+1]$ that are mapped to $\mathtt{G}_*$ or $\mathtt{G}_R$-type-2 pebbles (for some random coins s such that (1) is true). Then by Lemma 4.3 there exists a subset $\mathcal{S}' \subseteq \mathcal{S}$ of size $\sqrt{\bar{d}}/4$ such that the set of right parents $\mathcal{S}_R$ of $\mathcal{S}'$ is linearly independent over $\mathbb{Z}_2$; and for each gate $g \in \mathcal{S}'$ left and right parent are linearly independent. To see that the latter is true, note that any subset smaller than n of gates within one layer or within one column is linearly independent (cf. Lemma 4.2). It directly follows that left and right parents of any gate $g \in \mathcal{S}'$ since they lie in the same column. Furthermore, the set of left parents $\mathcal{S}_L$ to $\mathcal{S}'$ is linearly independent since it is a subset of $\leq \bar{d} < n$ gates at layer d.

To argue that D must have guessed many of the right input bits to $S^\wedge$ correctly, we use the following simple result from linear algebra. A proof can be found in the full version of this paper.

Claim. Let $m \in [1, n]$ and $\mathcal{S}_1 = \{u_i\}_{i\in[1,m]}$ a subset of $\{0,1\}^n$ that is linearly independent over $\mathbb{Z}_2$. Let $\mathcal{S}_2 = \{v_i\}_{i\in[1,m]}$ be a multiset of elements in $\{0,1\}^n$ such that $\mathcal{S}_2$ as a set is linearly independent over $\mathbb{Z}_2$. Furthermore, assume $\{u_i, v_i\}$ is linearly independent for all $i \in [1,m]$. Then there exists an index set $\mathcal{I} \subset [1,m]$ of size $|\mathcal{I}| = \lfloor m/4 \rfloor$ such that $\bigcup_{i\in\mathcal{I}}\{u_i\} \cup \{v_i\}$ is linearly independent.

Since the multiset $\mathcal{S}_L$ and the set $\mathcal{S}_R$ of left and right parents of $\mathcal{S}'$ are linearly independent (as sets), respectively, and for any $g \in \mathcal{S}'$ left and right input to $\mathtt{G}$ are linearly independent, we can apply the claim to obtain a subset $\mathcal{S}'' \subset \mathcal{S}'$ of size $|\mathcal{S}'|/4$ such that the union of the parents of $\mathcal{S}''$ is linearly independent. For $\mathcal{S}''$, we can now use Lemmas 4.7 and 4.8 to see that any successful D must have correctly guessed all right input bits to $\mathcal{S}''$; i.e., for s sampled uniformly at random, the probability that D succeeds is at most $(1/2)^{|\mathcal{S}''|}$. As $|\mathcal{S}'| \geq \sqrt{\bar{d}}/4$, the probability that D succeeds can be upper-bounded by $(1/2)^{\sqrt{\bar{d}}/16} < (3/4)^{\sqrt{\bar{d}}/7}$.

□

Lemma 4.11. *Let r be arbitrary coins such that case (2) is true. Then the probability (over uniformly random coins s) that the output of $\mathcal{A}(s)$ and $\mathcal{A}[c^*](s)$ differs after interaction with $\mathsf{D}(r, c^*)$ is at most $(3/4)^{\sqrt{d-3\bar{d}}/4}$.*

Proof. Recall that whenever the consistency check passes, each wire in $\tilde{\mathsf{G}}$ can be uniquely associated with two keys. Now, in case (2), for all but $\bar{d}$ wires in $G \setminus G^0$ the following holds: By Lemmas 4.6 and 4.8, for each bit running over the wire w in G, there exists at most one key associated with w in $\tilde{\mathsf{G}}^\oplus$ such that the AND gates with right input wire w is mapped to a "good" (W or $\mathtt{G}_R$-type-1) pebble, while for the other key associated to w it would be mapped to a "bad" pebble ($\mathtt{G}_L$ or $\mathtt{B}$). Note that in the latter case D immediately fails.

This allows us to map keys associated with wires in $\tilde{\mathsf{G}}^\oplus$ to bits, hence implies a mapping from $(\tilde{\mathsf{G}}, \tilde{x})$ to a circuit $\hat{\mathsf{G}}$ and input $\hat{x}$, where $\hat{\mathsf{G}}$ contains at most $3\bar{d}$ "undefined" gates (note, each internal wire effects 3 gates in $\mathsf{G}^\oplus$). Now, for D to succeed, it has to simulate $(\tilde{\mathsf{G}}, \tilde{x})$ such that at least $d' := d - 3\bar{d}$ "well-defined" gates in the circuit $\hat{\mathsf{G}}$ differ from XOR gates and $\hat{x} = x_0$. At the same time, all input and output wires of the well-defined gates have to carry the correct bits during evaluation (for "evaluation" of $\hat{\mathsf{G}}$ on $\hat{x}$ we apply the mapping from keys to bits to $\mathsf{Eval}(\tilde{\mathsf{G}}, \tilde{x})$ to extract a bit for all wires connected to well-defined gates).

Ignoring the undefined gates in $\hat{\mathsf{G}}$, this exactly corresponds to the game introduced in Section 4.2: D simulates a circuit such that all but d' gates are garbled correctly as XOR gates, and D succeeds, if for all gates the (input and) output bits correspond to the respective bits during evaluation of $\mathsf{G}^{\oplus}$ on input x_0. Lemma 4.1 now implies an upper bound on D's success probability in case (2): $\Pr[\mathsf{D} \text{ succeeds in case (2)}] \leq (3/4)^{\sqrt{d'}/4} = (3/4)^{\sqrt{d-3\bar{d}}/4}$. □

Thus, Lemmas 4.10 and 4.11 imply the following bound on any non-rewinding PPT distinguisher D (choose $\bar{d} = d/4$):

Corollary 4.2. *No non-rewinding PPT distinguisher* $\mathsf{D}^{\mathcal{F}}$ *can distinguish* $\mathcal{A}[c^*]$ *from* $\mathcal{A}$ *with probability larger than* $(3/4)^{\sqrt{d}/14}$.

To handle arbitrary – potentially rewinding – distinguishers D, we modify $\mathcal{A}$ as follows: Instead of sampling x_0, x_1 using random coins s, we assume a pseudorandom function f_k with uniformly random key k was hardcoded in $\mathcal{A}$, which takes as input a garbled circuit $\tilde{\mathsf{G}}$ and coins s, and outputs a tuple (x_0, x_1). Since D only has *black-box* access to $\mathcal{A}/\mathcal{A}[c^*]$, the secret key k is hidden from D, thus for two different inputs $(\tilde{\mathsf{G}}, s), (\tilde{\mathsf{G}}', s')$ to $\mathcal{A}/\mathcal{A}[c^*]$ the input pairs $(x_0, x_1), (x'_0, x'_1)$ look like independently sampled uniformly random strings.

With this modification in place, we finally arrive at the following lower bound on the security loss of any *black-box reduction* R (where we used $\delta < 3d$, hence $\sqrt{d}/14 > \sqrt{\delta}/25$). Note that our bounds naturally only apply to $d \leq n$, hence we assume $\delta < 2n$ in our theorem statement.

Theorem 4.1. *Any black-box reduction from the indistinguishability of Yao's garbling scheme (or its variant from [14]) on the class of circuits with input length* n *and depth* $\delta \leq 2n$ *to the IND-CPA security of the underlying encryption scheme loses at least a factor* $\frac{1}{q} \cdot (\frac{4}{3})^{\sqrt{\delta}/25} > \frac{1}{q} \cdot 2^{\sqrt{\delta}/61}$*, where* q *denotes the number of times the reduction rewinds the adversary.*

5 Discussion and Open Problems

In this work we prove that any black-box reduction from indistinguishability of (the modification [14] of) Yao's garbling scheme to IND-CPA security of the underlying encryption scheme must involve a loss in security that is sub-exponential in the depth of the circuit. This clearly also implies limitations to the stronger and more common simulation-based security and shows that the approach of [14] is essentially optimal. However, we leave it to future work if our fine-grained separation can be turned into an actual attack against Yao's garbling scheme.

Beside this most exciting open problem, one can also consider if our approach can be optimized. It might be possible to push our lower bound to an exponential loss, which would exactly match the upper bound from [14]. Following our approach, this requires a more sophisticated pebbling lower bound. Another interesting question would be if an even stronger bound can be found for the original construction of Yao, where the output mapping is sent in the offline phase, and certain limitations are already known from [3].

Acknowledgements. We would like to thank the anonymous reviewers of Crypto'21 whose detailed comments helped us considerably improve the presentation of the paper.

References

[1] P. Ananth and A. Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 455–472. Springer, Heidelberg, Nov. 2018.

[2] P. V. Ananth and A. Sahai. Functional encryption for turing machines. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, Jan. 2016.

[3] B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 166–184. Springer, Heidelberg, Aug. 2013.

[4] M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via UCEs. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Heidelberg, Aug. 2013.

[5] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In X. Wang and K. Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, Dec. 2012.

[6] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In T. Yu, G. Danezis, and V. D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, Oct. 2012.

[7] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

[8] S. Dziembowski, T. Kazana, and D. Wichs. Key-evolution schemes resilient to space-bounded leakage. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 335–353. Springer, Heidelberg, Aug. 2011.

[9] S. Garg and A. Srinivasan. Adaptively secure garbling with near optimal online complexity. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565. Springer, Heidelberg, Apr. / May 2018.

[10] B. Hemenway, Z. Jafargholi, R. Ostrovsky, A. Scafuro, and D. Wichs. Adaptively secure garbled circuits from one-way functions. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 149–178. Springer, Heidelberg, Aug. 2016.

[11] Z. Jafargholi, C. Kamath, K. Klein, I. Komargodski, K. Pietrzak, and D. Wichs. Be adaptive, avoid overcommitting. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 133–163. Springer, Heidelberg, Aug. 2017.

[12] Z. Jafargholi and S. Oechsner. Adaptive security of practical garbling schemes. In K. Bhargavan, E. Oswald, and M. Prabhakaran, editors, *Progress in Cryptology – INDOCRYPT 2020*, pages 741–762, Cham, 2020. Springer International Publishing.

[13] Z. Jafargholi, A. Scafuro, and D. Wichs. Adaptively indistinguishable garbled circuits. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 40–71. Springer, Heidelberg, Nov. 2017.

[14] Z. Jafargholi and D. Wichs. Adaptive security of Yao's garbled circuits. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 433–458. Springer, Heidelberg, Oct. / Nov. 2016.

[15] C. Kamath, K. Klein, K. Pietrzak, and M. Walter. On the cost of adaptivity in graph-based games. Cryptology ePrint Archive, Report 2021/059, 2021. https://eprint.iacr.org/2021/059.

[16] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, Apr. 2009.

[17] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, Nov. 1982.

[18] A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, Oct. 1986.